CHAPTER 1
THE OVERVIEW OF CHALLENGES AND PRACTICES IN SOFTWARE DEVELOPMENT TEAMS

Noorminshah A. Iahad and Masitah Ghazali

1.1 OVERVIEW

Software systems today are at once ubiquitous and mission-critical. From health monitoring and public services to education, mobility, and entertainment. Software mediates value creation in society. With this centrality comes a pressing need to improve how software is specified, designed, built, validated, deployed, and maintained. Importantly, these activities are not only technical—they are deeply social and organisational. Teams must navigate ambiguity, coordinate expertise, and align with institutional goals under time and resource constraints. As Sommerville (2016) argues, engineering quality is inseparable from process quality and management discipline. Persistent challenges such as requirements volatility, communication breakdowns, testing bottlenecks, and post-release support continue to surface across contexts (Menezes et al., 2019).

This book brings together practise-informed chapters from developers, analysts, and academics who have delivered complex software in the Malaysian research university context and beyond. The compilation does not prescribe a single methodology. Instead, it

spotlights concrete practises that teams have used to overcome real constraints: Blending User Experience (UX) in conventional lifecycles, modelling goals with stakeholder roles, institutionalising daily stand-ups, leveraging GitLab at scale, structuring agile delivery in healthcare, improving user acceptance testing (UAT), and professionalising helpdesk ticketing to close the loop between operations and development. Throughout, we emphasise actionable heuristics and patterns that complement established bodies of knowledge.

Chapter 1 serves as the book's roadmap. It situates the remaining chapters in the broader discourse of software engineering practise and management; synthesises recurrent challenges; presents a unifying lens to connect people, process, product, and platform concerns; and offers reading paths for different roles. The goal is to help readers grasp not merely what was done in each chapter, but why it worked, when to apply it, and how to adapt it in their own organisations.

1.2 HISTORICAL CONTEXT OF SOFTWARE ENGINEERING PRACTISES

The journey of software engineering has been shaped by successive attempts to bring discipline and predictability to an inherently complex and creative endeavour. Since the 1968 NATO Conference on Software Engineering, often regarded as the field's formal inception, the profession has evolved from viewing software as a product of individual craftsmanship to treating it as a managed, team-based process. Early paradigms emphasised structure and control: Methods such as Waterfall and V-Model formalised sequential phases of requirements, design, coding, testing, and maintenance. These approaches improved traceability but often failed to accommodate evolving requirements or stakeholder feedback.

By the 1990s, the community recognised the limitations of rigid life cycles. The rise of iterative and incremental models such as the Spiral Model, Rational Unified Process (RUP), and later Agile signalled a shift towards adaptability and collaboration. The Agile Manifesto (Beck et

al., 2001) further revolutionised thinking by prioritising individuals and interactions over processes and tools, and working software over comprehensive documentation. In parallel, the growth of open-source ecosystems and version control tools—CVS, Subversion, and later Git, reshaped how teams shared and reviewed code.

The 2010s brought DevOps, continuous delivery, and integrated toolchains that blurred boundaries between development and operations. In the same period, usability engineering and UX design entered mainstream software processes, enriching the discipline with human-centred perspectives. Today, software development in research universities, public agencies, and industry blends these influences: formal governance for accountability, agile practises for responsiveness, and human-centred design for usability. This historical evolution underscores a key idea: Effective software engineering is less about following a particular methodology than about adapting methods to purpose and context, a principle that anchors the chapters in this book.

1.3 METHODOLOGICAL LENS

This book adopts a practise-based research orientation, drawing upon lived experiences of teams as a legitimate source of knowledge. In contrast to controlled experiments or purely theoretical analysis, practise-based research privileges situated insight, how professionals navigate ambiguity, negotiate trade-offs, and apply judgment under real constraints. The focus is not merely on what practises exist but on how and why they work in particular contexts.

Each chapter contributes to this collective understanding by documenting interventions that improved coordination, quality, or user satisfaction within Malaysian research universities and related organisations. The contributors comprise academics, developers, analysts, and support staff who operate at the intersection of technology, education, and governance. Their reflections combine empirical observation with tacit expertise, what Schön (1983) termed "reflection-in-action".